Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 17 (2013) 545 – 553

Procedia Computer Science

Information Technology and Quantitative Management (ITQM2013)

A Middleware Architecture for Price Comparison Service on Mobile Phones

Gangmin Li[a]* Zhun Shen[b]

[a]Departement of Computer Science and Software Enineering, Xi'an Jiaotong-Liverpool University, 111 Renai Road, Dushu Lake Higher Education District, Suzhou 215123, China

[b]*IBM , 88 Dongchang Road Suzhou Industrial Park, Suzhou 215128, China*

Abstract

E-commerce solutions based on mobile devices have drawn great attention recently not only because it provides convenience to mobile users but also it generates revenue for service providers. However our study reveals that current applications failed in users' satisfactory because of three major issues: service availability, data accuracy and usability. Based on this discovery, we have proposed a solution, which is a middleware architecture that accommodates agents technology, service mash-up and a "contribute-reward" mechanism into a SOA. A model implementation has also been built for a mobile price comparison to test our proposed architecture and solutions. Results demonstrate the improvements on the issues addressed.

Selection and peer-review under responsibility of the organizers of the 2013 International Conference on Information Technology and Quantitative Management

Keywords: Mobile computing; e-commerce portal; middleware; usability;

1. Introduction

With the rapid development of mobile technologies, web services and communication technologies, many applications have been developed and deployed on mobile devices [1, 2, 3, 4]. One of the most popular application is E-commerce integrated with Geographic Information based Services (GIS) to enhance already existing real-estate shops or on-line shops, such as Taobao, Tmall, 360Buy, Tuan800** and many other similar applications [8, 5, 7, 6]. These applications have a few features in common: (1) they all extend their existing business to support mobile portal (have a mobile app to download); (2) they all integrate geographic information such as user's location and shops where the products available; (3) they all support rich user

* Corresponding author. Tel.: +1-391-541-9474.*E-mail address:* gmli64@gmail.com.

** Those are popular Chinese mobile applications availiable on Apple store, Android and Windows Mobile platforms.

Selection and peer-review under responsibility of the organizers of the 2013 International Conference on Information Technology and Quantitative Management

doi:10.1016/j.procs.2013.05.070

interface such as text or voice query; (4) they all support full e-commerce activities such as price comparison, products purchase and delivery. However our recent study shows that those applications failed to reach the expected level. Apart from social and cultural problems [10], we have identified the following technical problems: The first problem is the service availability. There are not only communication problems due to most mobile communication falls into "best-efforts" rather than QoS. So that a mobile client can loss connections with a server from time to time; there are also server "no-response" problems where a server may have no response or have a response with no data. The second problem is the date accuracy. Users stop using a service because the data provided by the service provider such as price of a product and available shops of a product are not accurate in ways of the data are either too old or incorrect. The third problem is the usability that is users' experience with the service, which mainly refers to the poorly designed interface and tedious operations. Existing problems prevent applications being used widely among the mobile users. This paper reports our efforts on solving these issues. The reset of the paper is arranged as follows: section 2 presents our proposed middleware architecture, which is used in our application and we believe it has a general implication for any similar applications. Section 3 describes the integrated technologies such as dynamic service selection, data harvesting and the "contribution-reward" mechanism. They are used to improve the data accuracy problem. Section 4 describes our model implementation of the middleware architecture and the newly introduced technologies in a mobile price comparison application. Section 5 reports our findings. We conclude that the proposed architecture and the technologies are useful in resolving the issues we identified in mobile geographic based service development and they can be used to improve mobile e-commerce solution in general.

2. Design requirements and main components of the middleware architecture

In this section, we discuss the main design requirements of the proposed architecture and present the main architectural components of the proposed middleware architecture.

2.1. Requirements

Most mobile portal for e-commerce applications adopts straightforwardly "client-server" architecture. Where, a mobile portal is deployed on a mobile device to act as a client. The business logic and operations are implemented and deployed on a dedicated server. Mobile client sends request through different mobile communication channels (2G or 3G) and the server response with data and service available. This basic structure is an instance of the service oriented architecture (SOA). It has advantages of easy implementation and deployment. It also addresses interoperability issues in a way that different type and format of data can be incorporated into one system with one unique presentation. SOA provides an efficient framework for distributed service communication regardless of the machine and languages used for service execution and description. However it suffers from the service availability problem in practices. This is because mobile communication is not reliable as landline networks. The connection is normally in a "best efforts" rather than ensured quality of service (QoS) fashion. To ensure mobile services as efficient and reliable, additional requirements are needed. One way of improve the connectivity and increase the service availability is to build redundant services with mash-up or dynamic service selection and integration. For example, to overcome the un-available of Taobao price service, an alternative connection is needed to the similar services provided by Tmall or other service providers. However this will require a structural change of the mobile client, which is normally not feasible in practice. The only way to include the multiple redundant services provider is building a mash-up on the server. The server will need not only provide the basic mush-up for multiple services but also resolve the data redundancy and conflict caused by the multiple service providers and multiple data sources. What is more, mash-in-time is not a good practice for a mobile application since the connection is costing and any delay of response will result in users' quitting of the connection. Therefore a cache mechanism is needed to support multiple data sources. In the architecture that we are proposing, it does not only provide a cache to

accommodate multiple data source and redundant service providers, but also supports multi-agent harvesting mechanism. Each agent is a web harvester specially designed to use a service provider's API to harvest product information needed such as price and shop location from the service provider. All the agents store the harvested data into a common cache where data redundancy and conflict will be resolved when new data is arrived.

2.2. Main architectural components and functions

The main architectural components of the proposed middleware architecture are presented in Fig. 1.

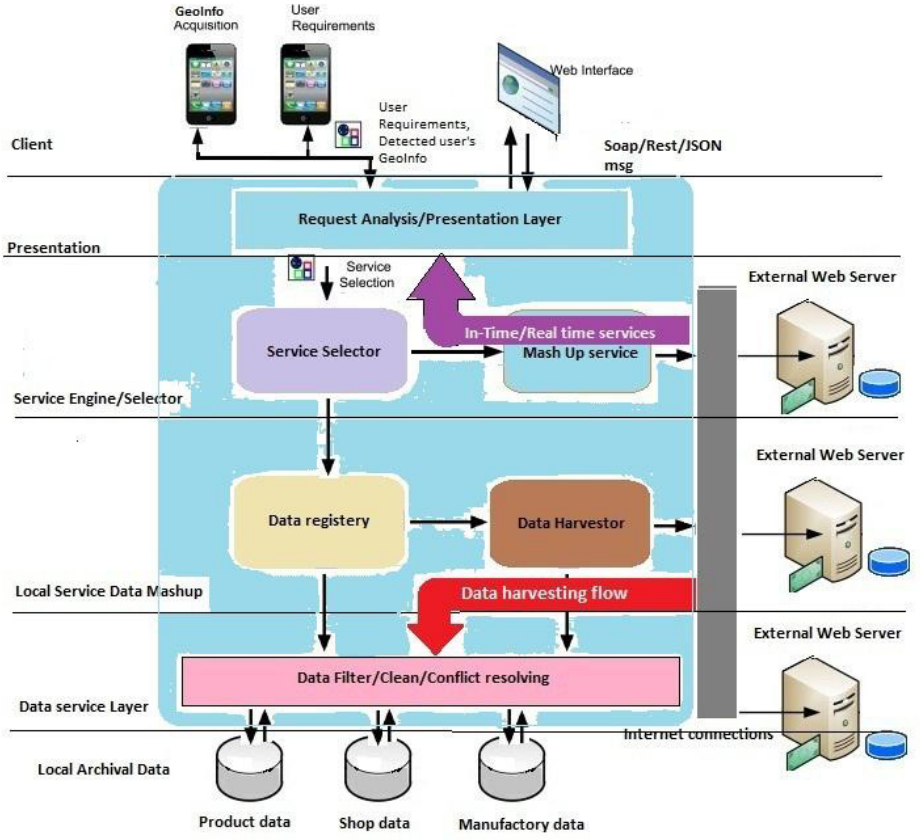

Fig. 1. Middleware Architecture with major components

In Fig 1, the proposed architecture is divided into four layers as a normal SOA does. However each layer has different functionality. From application to physical data storage the four layers are: Request analysis layer/result presentation layer, service mash-up layer, data mash-up layer and data filtering/cleaning layer.

Request analysis and result presentation layer is the most top layer near applications. Its major function is to separate users' query parameters with users' data contribution. For example a user may requests a product price with bar code of the product. Here bar code is the user's input, in the same time mobile portal can get a user's geographic location coordinates by call a GPS module installed in the user's mobile device. They can be sent with user's request SOAP message, in the same time the current product's price can be entered as the user's contribution to the system. The request analysis layer will separate the user's request with the user's

contribution and geographic location information. In the response the layer will also represent data retrieved from the service into the user's required format. It may include user's personal preference in terms of product category or locational restrictions.

The Service Layer is the core of the proposed architecture. The service engine is setting in this layer to do service selection. Service selection mainly refers to a function that enables a selection among available services to satisfy a user's service request. For example, if a user demands a price comparison, however the service registry shows that it is available in local service, then the local data service will be invoked and the results will retrieved and send to presentation layer. This is an ideal case, a lot of cases, a user's service request may not exist in the local service registry, in this case the request will send to Service Mash-up Service as indicated in the light-blue box in Fig 1. Until then the Mash-up service will select and call external service providers to response the user's request. Generally multiple external service providers will be invoked. The results will be mashed-up together and send to the presentation layer.

It is observable that this module will enhance service availability by duplicate external services locally. If a service is not available locally and external services are invoked, then the services will be registered in the local service registry and not only external service URI is recorded but also functionality and data will be cashed locally as well.

The third layer is the data mash-up layer. It is a supporting layer for the service layer. Services need data support. The data can be local data or data from external service providers. Once requested data is available locally, the JDBC connector will connects local databases and get required data. However, in case of the required data is not available locally, external services will be called to get data from external service providers. In this case data registry will record the newly obtained data and the data sources. Special designed agents will be created to monitor the data sources for any data updating. Agents also run periodically to harvest data from external service providers.

The lowest layer in our proposed architecture is the data processing layer. In this layer harvested data and user entered data are filtered by the products list. Redundant data are removed and data with different format are unified.

The core of the proposed architecture is the ability to aggregate static data harvesting, dynamic service invocation and users' contributions and to store these data in a local cached relational data store represented as a JDBC resultset, which will managed by the data registry and processed by the data filters. Most users' request will be responded by the local service, however the dynamic or real time services mash-up is also supported by the architecture.

The systems developed following this architecture will possess a feature of performance refinements as per external service invocation will enhance the local service ability. It fundamentally transforms demand driven remote service invocation into a steady local service invocation. Therefore service availability will be ensured.

3. Data aggregations and service invocations

3.1. Dynamic service selection

One of the distinct models in proposed architecture is the service selector and dynamic service mash-up. Ideally service should be available all the time and the service should have data available for responding a client's request. However in reality this is not always the case. Most time a service may not available or does not have sufficient data to response a client's request. In this case dynamic service selection will be very useful. See Fig 2, where a user's request arrives, the service selector first checks the local UDDI to see if the required service is available locally, if yes, it will go direct to the local service call. Otherwise it will search the local UDDI to find the registered external service providers and then send service request to remote service providers, then duplicate service into local service registry with service instance. Dynamic service invocation is similar with the server-side dynamic service mash-up in functionality [9].

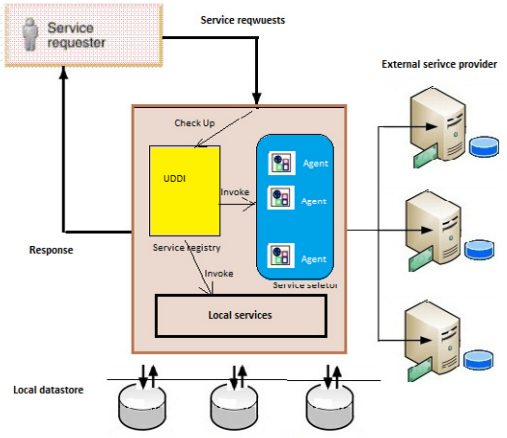

Fig. 2. Dynamic service selector

3.2. Data harvesting

Data harvesting is a model to maintain local data integration and accuracy. It is similar with data mash-up in web apps [9]. Once data to response a request is not available locally, there are two ways to get them. One way is to call external services and to get the data; the other way is suing agents. Special designed agents, each connect with one external service provider, are used to monitor the data source for any data updating. Once the data are changed then a harvest action will be performed by corresponding agent, so the local data store will have newly changed data. Agents also run periodically to harvest data from external service providers as shown in Fig 3.

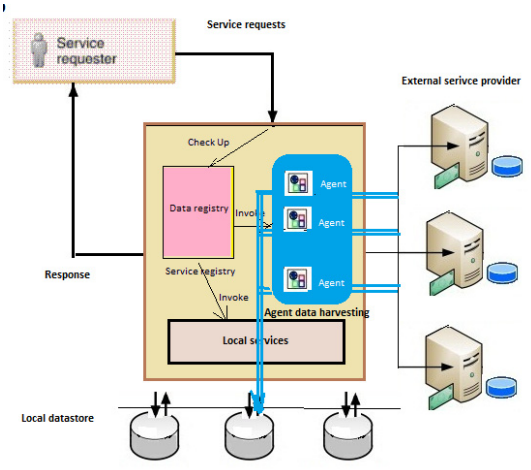

Fig. 3. Data Harvesting.

3.3. Contribution-Reward Mechanism

One of the most outstanding problems with mobile e-commerce application is the data accuracy. Although agents technologies are used to keep local data update automatically from remote data sources. Our study revealed that lacking of motivation for service provider to keep their date accurate either because their date is harvested from other service providers or because updating date request extra efforts and cost. A lot of service providers have real shops (as opposite to online shop as virtual shops). There is a tendency that they deliberatively lower the price on the web presents to draw customers into their real shops. However this data inaccuracy reduces the usage of the web portal installed on a mobile device.

Our study also reveals that due to the mobility most mobile user like to check product price where they are in front of a product. This provides a great opportunity for customs to become price input and verifying agents. However the cost of enter data and providing accurate data should not require too much efforts.

We have proposed the "Contribute-Reward" model to encourage users to provide data. The model works like this: in order to get a price comparison, the service users need to enter current price of a product. Only by doing so, the service will return prices of the same product in other shops. We call users' data input action as a "contribution" and service return as a "Reward" to users' efforts. Clearly, this data input should not be too expensive to put off users. The design of the portal for data input should focus on reducing the cost of users' data entry. Many methods can be adopted to do so. For example, bar code scan, voice enter, and even picture recognition etc.. Geographic information can be obtained from a GPS module on most mobile devices. So the users; input data can be easily associate with the geographic information.

Two issues related with the service return need to be clarified in here. One is regarded with so called the same product. Finding an identical product of a product in practice is not always feasible and desirable since in most cases different manufactures have different packaging for the similar products. In this case, returned products from the service are listed from identical to similar contents in three categories: same products, similar products and desirable products. The other issue is the available shops. Users care less about shops too far away. A user only desires price comparison with the shops either frequently visited or geographically within reachable distance.

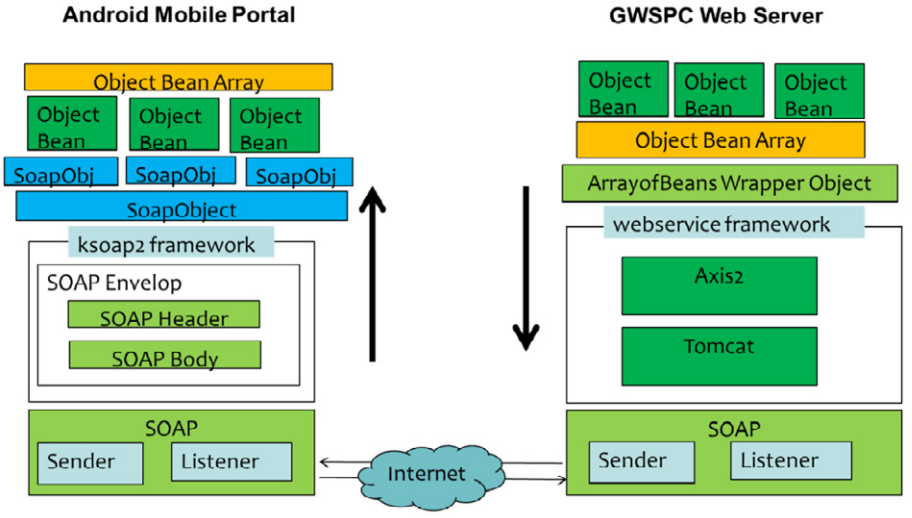

Fig. 4. Technical components of GWSPC mobile system.

4. Implementation and development of a mobile price comparison service

A real world application has been implemented. The application provides a "Geographic Based Web Service Portal for Price Comparison" (GWSPC). GWSPC adopts the service oriented architecture discussed above and the "contribution-award" mechanism is used to collect data from users and provide services for the users. Fig 4 shows the technical components of the implementation.

Fig. 5. Screen Captures of GWSPC on a mobile phone. (a) The user interface of the GWSPC portal; (b)bar code scanner; (c)Voice input interface; (d) Text input interface; (e) Geographic location display; and (f) service returns.

In this implementation the client is a mobile portal which is in APK form. The mobile portal can be downloaded and installed on a mobile devices running Android. Our mobile portals can provide a "price comparison" service based on input queries and users' geographic location. Server response includes price, available shops and location of the shops on matched product, similar products and products that users may interested with. Users' query can be submitted in various methods include text, voice, and bar code scanning. It is on the client we implemented the "contribute-reward" mechanism, where a user needs to provide current price in order to get a price comparison.

Proposed server architecture is implemented on a tomcat server running Axis2 SOAP engine, database is incorporate with server through JDBC to utilize the local service and data cache. Multiple agents are implemented to connect with external service providers to harvesting products' data.

GWSPC has been installed on an Android based mobile phone (MOTOROLA). Fig 5 shows some screen captures. The major interface is shown in a. Screen capture displayed in b is the barcode scanner. It will be the major query input method. C shows the other user query method with is the voice recognition interface. For users who don't like type in tiny keypad or do not know Chinese Pinyin (Chinese characters are typed in mostly using Pinyin), voice is the most useful way to enter queries. Screen capture d shows the supplementary information to barcode and voice can be entered by the normal text box. It is also used to enter a query in keywords or phrases. Screen capture e shows the current location. It can also display location of shops nearby. Screen capture f shows the results retuned by the price comparison service.

5. Tests and Results

A number of tests have been carried out. The following test results were obtained:

- GWSPC running on Android platform performs well in Robotium Test [11] and Android Lint Test[12]. All the widgets of mobile client perform as expected.
- Web service (through GWSPC) test results are displayed in table 1 below. Where 1000 requests with various response time were issued from GWSPC portal and success response were recorded with response time. The table indicate that 90% of the requests are responded in 600ms and more than 50% of the requests are responded in 95ms.
- From table 1, the response time between 90% of the success response to 100% of the success response has been increased dramatically. The reason is that the web server is designed can only handle maximum 1000 requests at a time. We can see that when the number of request and the response time of each request increase, the total response time for request completion increase sharply. The result shows that the overall performance, apart from the mechanism we introduced in our study, the server performance is also an important factor. In our implementation, to keep high performance of the server, the throughput should less than 900.

Table 1. Completion times for percentage of request get response successfully. 1000 requests, each with different response time, were issued on GWSPC.

Percentage of success response from server (%)	50	60	70	80	90	100
Time taken to complete requests (ms)	95	103	221	302	673	22094

6. Conclusion

Design and implementation of a mobile price comparison service is challenging in real practice. Apart from cultural and social problems, technical problems can stop users to use the application. Applications deployed on a mobile device meant to use its mobility to provide users with easy access and in-time service when needed. However our study shows that current ecommerce solutions on mobile device fall away from its expectations. Three major issues we have identified are: service availability, data accuracy and user experience. To overcome problems caused by these issues requires multiple efforts in multi-disciplinary area, involving mobile communication, system design and software engineering. Real time service in distributed, heterogeneous systems and service providers is a challenging task itself.

In this paper, we proposed a service oriented architecture for mobile ecommerce solutions, incorporating service selector, data harvester and presentation services, as well as dynamically external service and data local caching. Furthermore, the proposed architecture exploits added-value services like data filtering, agent auto data updating and user's "contribution-reward" model able to provide services in real time and increased data accuracy. A sample implementation on mobile price comparison has been developed using proposed architecture to verify its practical usability. Preliminary test demonstrates its effectiveness on resolving general issues discovered.

The main contribution of this paper is focused on resolving issues discovered in current mobile application study. The proposed architectural integrated with the dynamic service selection, data harvesting agents and "contribution-reward" model allows heterogeneous data sources, presentation frameworks and added-value service providers to be integrated in a coherent system in order to provide steady service and accurate data to mobile users.

Acknowledgements

Thanks the SURF programme of XJTLU for providing funding for this research and development.

References

[1] Minder Chen, Dongsong Zhang, Lina Zhou, "Providing Web Services to Mobile Users: The Architecture Design of an M-Service Portal",International Journal of Mobile Communications, Vol. 3, No. 1, 2005, pp. 1-18.
[2] Robert B. Doorenbos , Oren Etzioni , Daniel S. Weld, A scalable comparison-shopping agent for the World-Wide Web, Proceedings of the first international conference on Autonomous agents, p.39-48, February 05-08, 1997, Marina del Rey, California, United States
[3] Cik Ku Haroswati Che Ku Yahaya(2011).A Framework on Halal Product Recognition System through Smartphone Authentication, Advances in Automation and Robotics,Vol.1, 2011 Springer-Verlag Berlin Heidelberg.
[4] Konstantinos Tserpes , Fotis Aisopos , Dimosthenis Kyriazis , Theodora Varvarigou A recommender mechanism for service selection in service-oriented environments. Future Generation Computer Systems, Volume 28, Issue 8, October 2012, Pages 1285–1294.
[5] Vassilios Vescoukis, Nikolaos Doulamis, Sofia KaragiorgouA service oriented architecture for decision support systems in environmental crisis managementFuture Generation Computer Systems, Volume 28, Issue 3, March 2012, Pages 593–604.
[6] Wojciechowski, A., Supporting Social Networks by Event-Driven Mobile Notification Services. In: Meersman, R. (ed.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 398–406. Springer, Heidelberg (2007).
[7] Lee, K.J., Lee, J.C., Design of Ubiquitous Referral Marketing: A Business Model and Method. In: Bauknecht, K., et al. (eds.) EC-Web 2006. LNCS, vol. 4082, pp. 102–111. Springer, Heidelberg (2006).
[8] Story, L.: New Bar Codes Can Talk With Your Cellphone, New York Times (April 1, 2007).
[9] Ahimanikya Satapathy, Srinivasan Rengarajan, Karthik S, Using the Enterprise Data Mashup Service Engine. Part No: 820-7856-10 Sun Microsystems, Inc. 4150 Network Circle Santa Clara, CA 95054 U.S.A.
[10] Hiltunen, M., Laukka, M.and Luomala, J. 2002. Mobile User Experience, Edita Publishing Inc. Finland, 214.
[11] Renasreda,2011.Robotium,viewed 21/21/2012,from http://code.google.com/p/robotium/
[12] Tor Norbye, Nov 15, 2011. Android Lint ,viewed 21/12/2012, from http://tools.android.com/tips/lint